# DOM (Document Object Model)

* D = Document in HTML form
* O = Every Element / object in HTML
* M = hierarchy / objects in a structural way

# DOM Tree Model



# DOM Elements / Objects Styling

**Note: use “ ” instead of : or –**

**Note2: remember style = “ ” use as attribute name pair but = between them used in html**

**Note3: For DOM we use attribute.attribute = “name”**

**Note4: use : and ; for name properties only and in HTML**

**Following example represent note 2 and note 3**

<!-- in html we use -->

<p style="border: 2px solid black;" ></p>

//for DOM

let El = document.querySelector(".xyz").style.color = "red"

# Targeting DOM Elements / Objects Tag, class, Id

* getElementById()
* querySelector()
* getElementsByClassName()
* getElementsByTagName()
* querySelectorAll()

Note: first we targeting DOM Elements then can inner and get or set attributes

<p id="abc" class="xyz xyz2" style="border: 2px solid black;" >paragraph </p>

<script>

//targeting DOM Elements or Object by considering the above condition

//we use get and set Attributes with them

let El\_id = document.getElementById("abc")

console.log(El\_id); // give the id value + class value

let El\_querty = document.querySelector(".xyz")

console.log(El\_querty); //return simple classes

//below 2 getClass and getTage works the same

let El\_class = document.getElementsByClassName("xyz")

console.log(El\_class); //return array consist all classes

let El\_element = document.getElementsByTagName("p")

console.log(El\_element); //return array consist all classes

let El\_quertyAll = document.querySelectorAll(".xyz")

console.log(El\_quertyAll); //return array consist all classes

# DOM Elements / Objects => Text

let El = document.querySelector("p").innerText

console.log(El); //only his Text

let El2 = document.querySelector("p").innerHTML

console.log(El2); //All Tags + Text

let El3 = document.querySelector("p").textContent

console.log(El3); //Give his and his children Text

# DOM Elements / Objects => get-Attribute

<h1 id="aaa" style="height:2px; color: aquamarine;"></h1>

let El = document.querySelector("#aaa").getAttribute("style")

console.log(El); //return attribute value pair

let El2 = document.querySelector("#aaa").attributes[1].value

console.log(El2); //return all attributes in object form

# DOM Elements / Objects Set-Attribute method

<style>

        .reg{

            font-size: 100px;

        }

    </style>

<h1 id="aaa" style=" color: aquamarine;" class="reg" onclick="a()">sdsdsd</h1>

<script>

    let el = document.querySelector("#aaa").setAttribute("class","reg")

    let el2 = document.querySelector("#aaa").setAttribute("style","border:10px dotted yellow;")

let el = document.querySelector("#aaa").removeAttribute("style")

</script>

# DOM Elements / Objects classList methods

* classList
* classList.add(“class1”, “class2”,……..)
* classList.remove(“class1”, “class2”,……..)
* classList.toggle(“class”)
* classList.constains(“class”)

# DOM Elements / Objects Traversing

**Traversing is like traveling**

const Inner = document.querySelector(".inner")

//return parent information

console.log(Inner.parentElement);

//return array consist all children

console.log(Inner.children);

//return first child

console.log(Inner.firstElementChild);

//return last child

console.log(Inner.lastElementChild);

//return next sibling

const B = document.querySelector(".b")

console.log(B.nextElementSibling);

//return previous sibling

console.log(B.previousElementSibling);

//now useless things

console.log(Inner.parentNode);

console.log(Inner.childNodes);

console.log(Inner.firstChild);

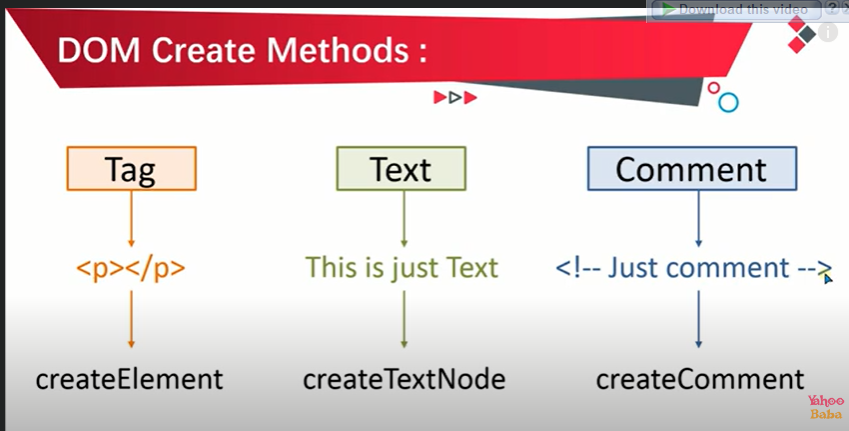
console.log(Inner.lastChild);

console.log(Inner.nextSibling);

console.log(Inner.previousSibling);

# DOM Create Method

1. createElement (for tag)
2. createTextNode (for Text)
3. createComment (for Comments)



//DOM Create Methods

//createElement

//createTextNode

//createComment

const newElement = document.createElement("h2")

console.log(newElement); // <h2></h2>

const newText = document.createTextNode("text inside tage")

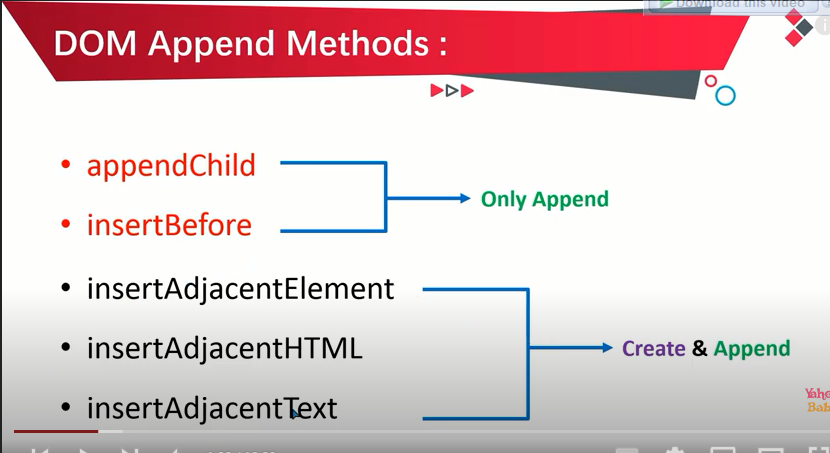
console.log(newText); // "text inside tage"

const newComment = document.createComment("just for testing")

console.log(newComment); // <!--just for testing-->

# DOM append methods

1. appendChild()
2. insertBefore()
3. insertAdjacentElement()
4. insertAdjacentHTML() no need of append method
5. insertAdjacentText() no need of append method



1. appendChild() ()

//DOM append Methods

//appendChild

//////////////////////////////////////////

const Div = document.querySelector(".inner")

const newElement = document.createElement("h2")

console.log(newElement); // <h2></h2>

const newText = document.createTextNode("text inside tage")

console.log(newText); // "text inside tage"

const newComment = document.createComment("just for testing")

console.log(newComment); // <!--just for testing-->

////////////////////////////////////////////

//1.appendChild (append at last)

newElement.appendChild(newText)

console.log(newElement); // <h2>"text inside tage"</h2>

Div.appendChild(newElement) //text in div

1. insertBefore (may use in future so skip it here)

//DOM append Methods

//insertBefore

//////////////////////////////////////////

const Div = document.querySelector(".inner")

const newElement = document.createElement("h2")

console.log(newElement); // <h2></h2>

const newText = document.createTextNode("text inside tage")

console.log(newText); // "text inside tage"

const newComment = document.createComment("just for testing")

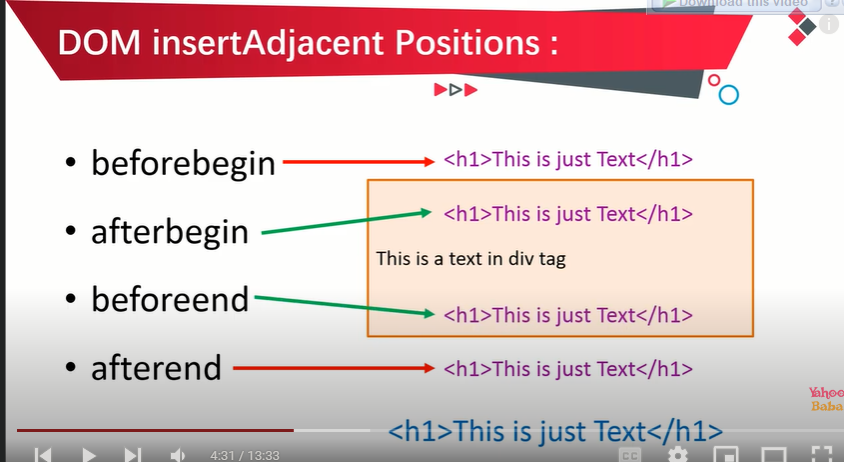
console.log(newComment); // <!--just for testing-->

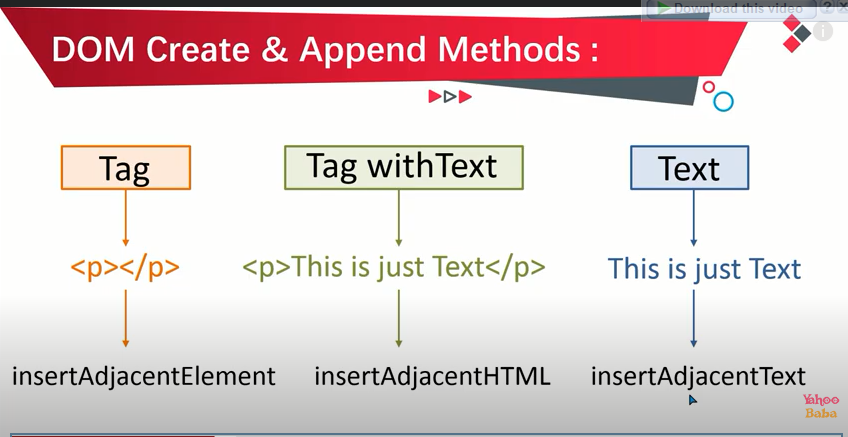
////////////////////////////////////////////

//2. insertBefore (append freely) not recomended

Div.insertBefore(newElement, Div.children[1])

console.log(Div); // <h2>"text inside tage at 1 index"</h2>





1. insertAdjacentElement()

//DOM append Methods

//insertAdjacentElement

//////////////////////////////////////////

const Div = document.querySelector(".inner")

const newElement = document.createElement("h2")

console.log(newElement); // <h2></h2>

const newText = document.createTextNode("text inside tage")

console.log(newText); // "text inside tage"

const newComment = document.createComment("just for testing")

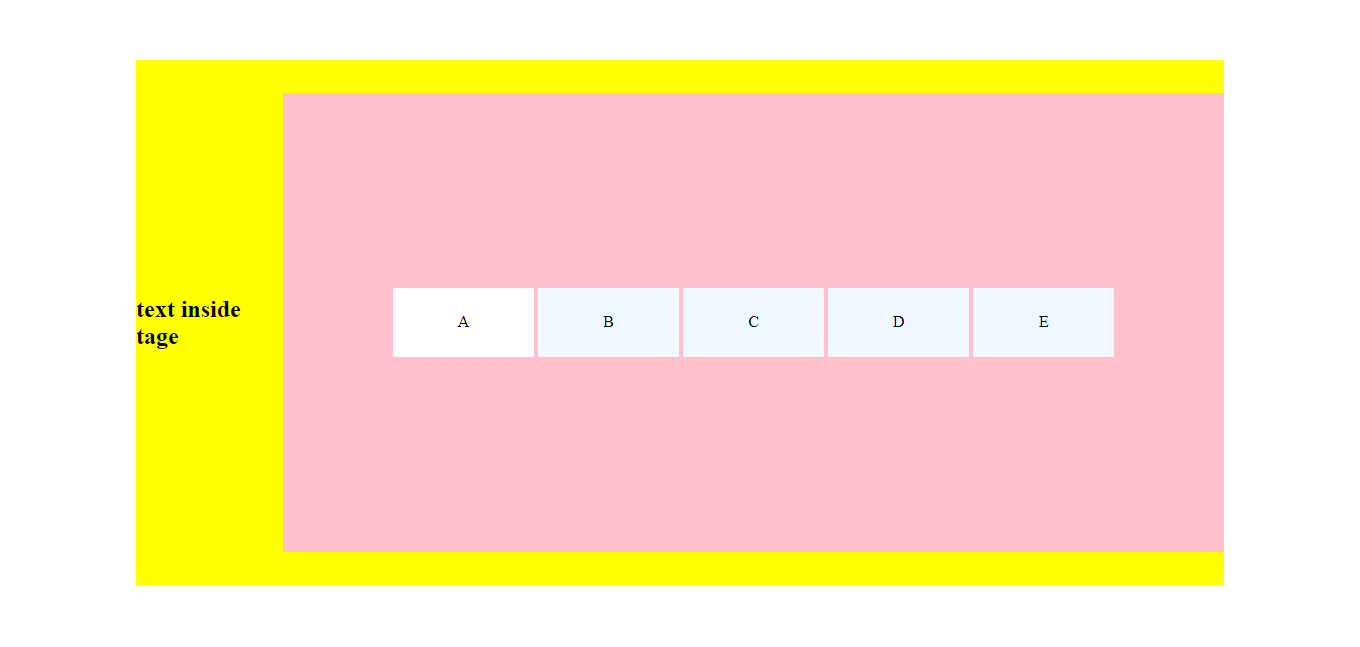
console.log(newComment); // <!--just for testing-->

////////////////////////////////////////////

// 3.insetAdjacentElement

newElement.appendChild(newText) //<h2>text inside tage</2>

Div.insertAdjacentElement("beforebegin",newElement)



1. insertAdjacentHTML

//DOM append Methods

//insertAdjacentHTML

//////////////////////////////////////////

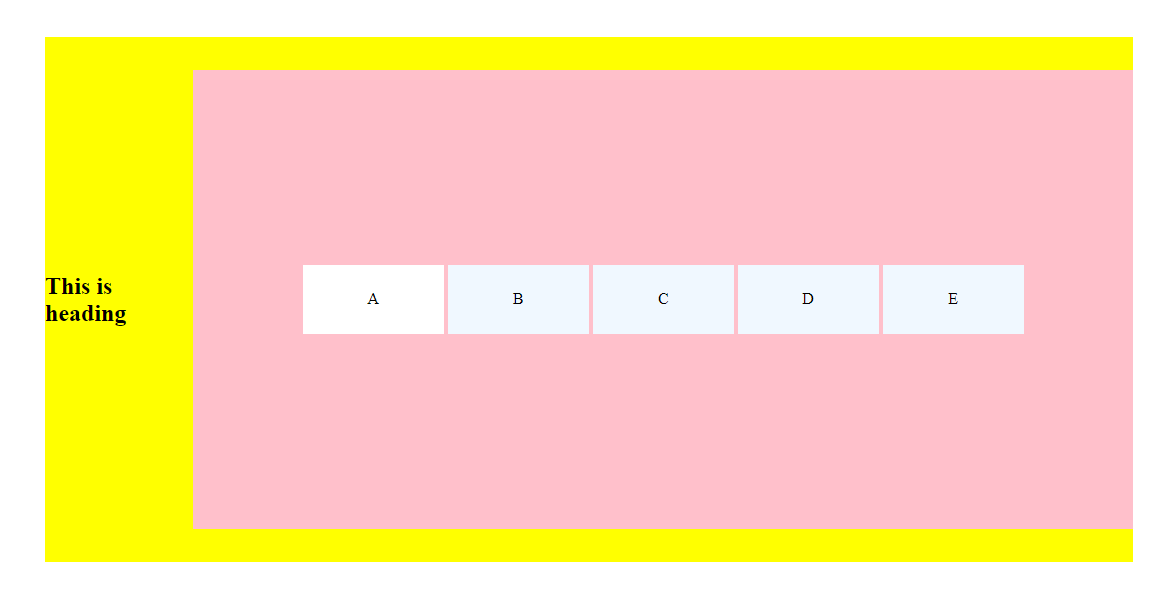
const Div = document.querySelector(".inner")

const newElement = "<h2>This is heading</h2>"

////////////////////////////////////////////

// 3.insetAdjacentHTML

Div.insertAdjacentHTML("afterend",newElement)



1. insertAdjacentText

//DOM append Mehtod

//insertAdjacentText

//////////////////////////////////////////

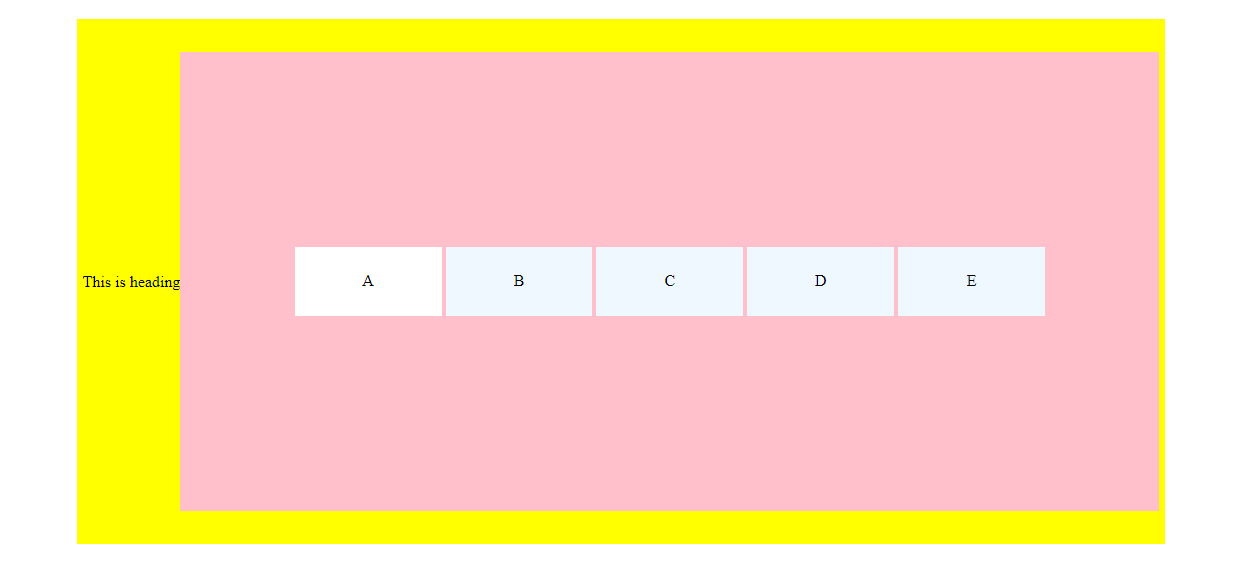
const Div = document.querySelector(".outer")

const newElement = "This is heading"

////////////////////////////////////////////

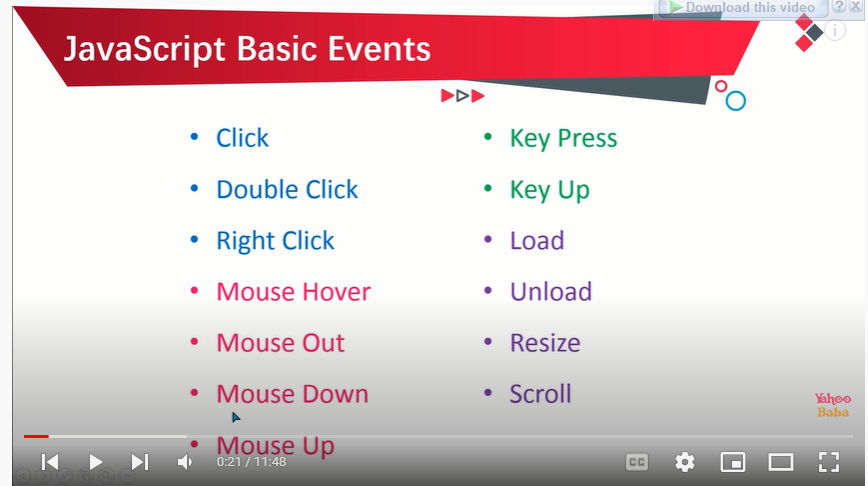
// 3.insetAdjacentText

Div.insertAdjacentText("afterbegin",newElement)

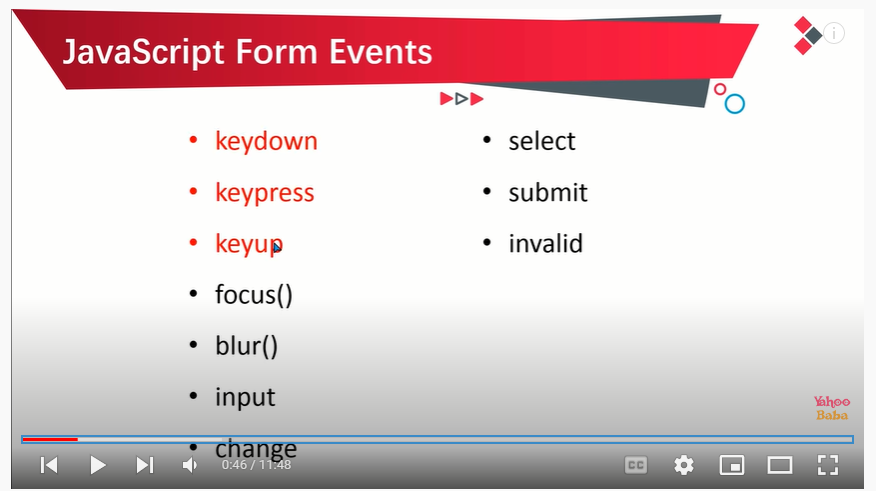


# Events

These are simple events but



# Form Events



# Form Events important examples

1. focus & blur

<body>

<form action="">

<label for="">name</label>

<input type="text" id="fname"

  onfocus="onfoc(this)"

  onblur="blurr(this)">

<label for="">class</label>

<input type="text"  id="lname"

  onfocus="onfoc(this)"

  onblur="blurr(this)">

</form>

<script>

function onfoc(e){

e.style.backgroundColor = "red"

}

function blurr(e){

e.style.backgroundColor = ""

}

* Very important fact is here this on html tag represent object, tag, or element of selected html tag\
* Focus and blur are opposite in nature

1. Input

<form action="">

<label for="">name</label>

<input type="text" id="fname"

    oninput="oninp(this)"

    >

<label for="">class</label>

<input type="text"  id="lname"

    oninput="oninp(this)"

    >

</form>

<div class="example"></div>

<script>

function oninp(e){

document.querySelector(".example")

.innerHTML = e.value

}

* Very important concept uses here is that this refers to object and in function e.value represent the input field value

1. Onchange

* Very important concept when we type or select something + press enter or lost focus onchange called
* Basically type and press enter

<form action="">

<label for="">name</label>

<input type="text" id="fname"

    onchange="onchang(this)"

    >

<label for="">class</label>

<input type="text"  id="lname"

    onchange="onchang(this)"

    >

</form>

<div class="example"></div>

<script>

function onchang(e){

document.querySelector(".example")

.innerHTML = e.value

}

1. On submit Event

* Very important thing is that onsubmit() does not apply on submit button it apply on form tag

<form action="" onsubmit="onsub(this)">

<label for="">name</label>

<input type="text" id="fname"

    onchange="onchang(this)"

    >

<label for="">class</label>

<input type="text"  id="lname"

    onchange="onchang(this)"

    >

<input type="submit" name="" id="sub">

</form>

<div class="example"></div>

<script>

function onsub(){

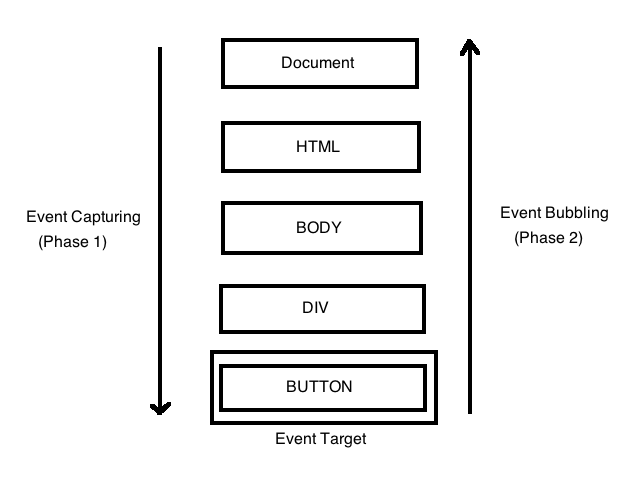
    let x = document.querySelector("#fname").value;

     alert(`hello ${x}`)

}

# Event Propagation

* Event propagation is **a way to describe the “stack” of events that are fired in a web browser.** Which means on click on child must fire child to grand-grand-parent event also
* By default, browser perform bubbling on events
* If third argument in addEventListner is true then the phase will be capturing



Bubbling (upward Propagation)

let g = window.document.querySelector(".grandParent")

let p = window.document.querySelector(".parent")

let c = window.document.querySelector(".child")

g.addEventListener("click",()=>{

console.log("grandparent");

},false)

p.addEventListener("click",()=>{

  console.log("parent");

},false)

c.addEventListener("click",()=>{

  console.log("child");

},false)

Capturing (Downword Propagation)

let g = window.document.querySelector(".grandParent")

let p = window.document.querySelector(".parent")

let c = window.document.querySelector(".child")

g.addEventListener("click",(e)=>{

console.log("grandparent");

},true)

p.addEventListener("click",(e)=>{

  console.log("parent");

},true)

c.addEventListener("click",(e)=>{

  console.log("child");

},true)

# Event Delegation

* Instead of adding an event listener to each and every similar element, we can add an event listener to a parent element and call an event on a particular target using the **target** property of the event object.
* **Target** hold all information about **click**
* Here target = div.clicked area

let g = window.document.querySelector(".grandParent")

let p = window.document.querySelector(".parent")

let c = window.document.querySelector(".child")

g.addEventListener("click",(e)=>{

console.log(e);

},false)

Example 2

let g = window.document.querySelector(".grandParent")

let p = window.document.querySelector(".parent")

let c = window.document.querySelector(".child")

g.addEventListener("click",(e)=>{

let target = e.target

if (target.matches("li")) {

  target.style.fontSize="4rem"

} else {

  target.style.backgroundColor= "red"

}

console.log(e);

},false)